Design of a General Clinical Notification System Based on the Publish-Subscribe Paradigm

Antoine Geissbühler, M.D., Jonathan F. Grande, Randall A. Bates,
Randolph A. Miller, M.D., William W. Stead, M.D.
Informatics Center
Vanderbilt University Medical Center
Nashville, Tennessee

We describe the design and initial implementation of a notification sub-system, as a component of a modern information management architecture. The system, based on the publish-subscribe paradigm, provides a framework of event-based communications for the implementation of various important clinical applications including the notification of alerts and reminders with escalation algorithms, the reliable distribution of documents, and the implementation of intelligent patient-specific monitoring processes. The initial implementation of the system, providing the notification of the unit staff about new orders, indicates that the model is viable both in terms of functionality and ability to scale up.

INTRODUCTION

The ability to notify users and applications of new events represents an essential function of a patient care information system. Most traditional systems rely on the demand-driven (or request-reply) model to query or poll databases for new data. Several event-based clinical systems have been developed [1,2] and proved successful at notifying physicians and nurses of critical lab values [3,4,5] and reminders [6]. However, the ubiquitous presence of asynchronous events in clinical data-processing warrants the elaboration of a general notification system as one of the core components of an information management architecture.

Developments in the field of distributed event-based computing have produced a communications model, known as the publish-subscribe paradigm, which enables decoupled and asynchronous communication between distributed applications. This model addresses limitations of the request-reply model, in particular those related to the ability to handle events, the addressing of messages and scalability issues in an heterogeneous, distributed environment. Several commercial products support this communications model, and standards such as OMG CORBA [7] use it as a base for their event services specifications.

The goals of this project were to investigate how this communications model could be used and adapted to provide a general framework for the implementation of clinical event-driven functions in a large health center, and to build a working prototype to investigate the scalability of the model.

BACKGROUND

The publish-subscribe paradigm

The publish-subscribe paradigm provides a framework for the exchange of data between independent applications in an event-driven manner, by decoupling the sources of information (publishers) from its consumers (subscribers). Publishers anonymously broadcast messages onto the network and subscribers anonymously receive messages without having to request them, unlike the traditional request-reply communications model, where each request must be queued and answered individually.

Instead of actual addresses (or the coupling of specific applications), messages are classified by subjects which describe their contents and/or logical destination in a uniform name space. Subscribers independently look for subject labels of interest, without regard for the source of the message. The subject for an abnormal serum potassium level for patient 12345678 would be:

```
lab.mr#12345678.serumK.abnormal
```

a message alerting the surgical ICU of a mass casualty:

```
alert.SICU.massCasualty
```

where the subject describes an event and its parameters following a pre-defined syntax. For example, based on the following syntax:

```
admission.team number.nurses station
```

the subject:

```
admission.blue3.9South
```

refers to a new admission for team blue-3 in nurses station 9-South. A physician in team blue-3 wanting to be notified about laboratory results for his patients and admissions to his team would subscribe to:

```
lab.mr#12345678.*
lab.mr#23456789.*
admission.blue3.*
```

A rules engine checking potassium levels would subscribe to:

```
lab.*.serumK.*
```

A medical receptionist's workstation in SICU would subscribe to:

```
admission.*.SICU
alert.SICU.*
```

VUMC's environment

Vanderbilt University Medical Center's (VUMC) patient care information system is built on the MCIS-1 information system architecture [8] which supports distributed applications and integrated databases. From the notification system's design perspective, the key aspect of this architecture is the communication subsystem known as the Generic Interface Engine, which mediates the exchange of information between applications, and provides access to institutional databases.

DESIDERATA AND SYSTEM DESIGN

Desired capabilities

We identified a set of required functionalities for the general notification system, including:

- the distribution of patient-specific events such as new lab results, new orders, bedside monitor alarms, with the ability to handle alternative notification methods (escalation algorithms) in urgent situations such as critical lab values or STAT order, so that if the one notification mechanism is not successful (i.e., there is no acknowledgment of the notification) within a critical time period, alternative, more aggressive notification mechanisms will be pursued;

- the reliable distribution of printouts, such as laboratory requisitions, with the ability to re-route to backup printers in the case of localized network or hardware failure;

- the communication of localized or hospital-wide messages such as a warning of mass casualty, or technical information about the availability of applications;

- the support of distributed applications for the monitoring of specific activities. An example of such an application, which could be "ordered" by a physician, would be to implement this plan: "*patient X is to be anticoagulated with IV heparin: please notify the nurse of each new PTT value and recommend adjustements based on protocol Y; notify the ordering physician if the PTT value is out of the range specified*". Such an application would both function as a subscriber (to lab values) and as a publisher (of recommendations and alerts).

Guaranteed delivery

The publish-subscribe paradigm is characterized by the anonymous nature of the communication, where both publishers and subscribers are unaware of each other.

However, in certain situations, it might be desirable to guarantee that some subscribers have received a message. For example, the notification that an urgent order has been placed for a patient might be received by several subscribers, but, if one of them acknowledges the message, the notification should be considered taken care of, and canceled from the other subscribers.

Escalation algorithms

The fact that messages can be acknowledged by subscribers also enables the implementation of escalation algorithms, i.e., the use of alternative methods of notification if the preferred one fails. For example, if an urgent order is not acknowledged within three minutes when a message is displayed on the computer screens on the patient's floor, the charge nurse could be paged.

To avoid unnecessary delays, the system can determine if the message to be published has been subscribed to, and, if not, will immediately use an alternative notification method: if no workstation is available to display the message about the new order, the charge nurse should be paged without delay.

Notifications and notices

The use of escalation algorithms implies that the notification system controls in some way the method of notification, which contradicts the notion that a pure publish-subscribe system should dissociate its publishers from its subscribers. To maintain this separation, which is desirable, we distinguish two entities: the notification and the notice.

The *notification* is the message generated by a publisher, which makes no assumptions on how the message will be delivered. The notification system receives the notification, assigns a notification algorithm based on the message's content, and executes the algorithm which will produce one or more notices.

A *notice* is an attempt to deliver the message. It is represented by the message itself, and by a subject which is assigned by the notification algorithm. The notice is considered delivered when an acknowledgment is returned by the subscriber, either as a result of an interaction with a user (e.g., the message has been seen and the user accepts the responsibility of taking care of it) or by an application (e.g., a fax has been successfully sent). Notices can also be associated with an expiration date and time. The acknowledgment or expiration of a notice will trigger a change in the state of the notification algorithm and several actions can be taken: the current notice can be removed from the publishable domain, a new notice can be added, or the notification can be marked as delivered and the notification process stops. As subscribers have no awareness of each other, the removal of a notice from the publishable domain is an active process, i.e., a notice to remove a notice.

System architecture

The notification system is composed of five processes (Fig. 1):

- the Generic Interface Engine (GIE) is VUMC's central communication hub to which publishers will send the messages;

- the Notification Engine Speaker receives notifications from the GIE and invokes the appropriate Notification Logic Module to generate notices, then distributes them to all Notification Engine Listeners; it maintain a list of active notifications and their associated notices, and forwards events such as notices acknowledgments and expirations to the Notification Logic Module; it also maintains a central database of notifications, mediated by the GIE, for recoverability purposes;

- the Notification Logic Modules provide the message-specific algorithms for generating notices in response to events generated by the Notification Engine Speaker;

- the Notification Engine Listeners match the subject of the notices to the subscriptions of their clients and distribute the relevant messages;

- the Notification Engine Clients are applications that subscribe to certain subjects and provide either a user interface or an application interface to handle notices and, when appropriate, to acknowledge them.

This layered architecture allows to distribute the burden of data processing while guaranteeing recoverability from a central database. As only the relevant data is transmitted from the Listeners to the Clients, network traffic is minimized.

There is also a clear delimitation between the transport mechanism (i.e., the GIE, the Speaker and the Listeners), the message-dependent logic (i.e., the Notification Logic Modules) and the subscribing applications (i.e., the Clients).

CURRENT IMPLEMENTATION

Clinical applications at VUMC include MARS [9], an integrated browser for lab results and text reports and WizOrder, a care provider order entry system [10].

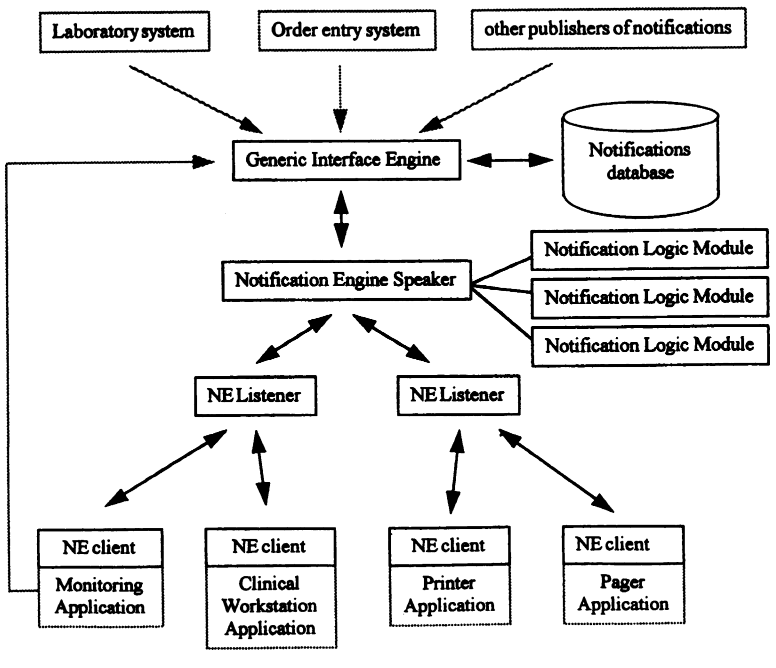

Figure 1. Notification System's Architecture

MARS is available throughout the institution, WizOrder is currently implemented for inpatients in medicine, surgery and OB/GYN.

Notification of orders

We decided to test the concept of an event-based communication model for the notification of the unit staff about new physicians' orders. Several reasons guided our choice:

- at the unit staff level, the current mechanism of notification for new physicians' orders is based on printouts, which are prone to hardware problems and manipulation errors such as erroneous discarding or misplacement in charts; moreover, there is no reliable way to document the fact that a care-giver is aware of the presence of new orders;

- several thousands of notifications are generated from new orders each day and their different urgencies suggest various notification algorithms and mechanisms; this would provide enough volume and diversity to test the scalability of the system.

The Clinical Workstation Client

The main client application of the notification system runs on the clinical workstations which are used for patient tracking, order entry, results reporting, e-mail and other functions. An animated map of the unit replaces the screen saver, and shows, with color codes, which beds have pending orders, and how urgent these are. A visual clue indicates the status of patients in beds close to the workstation. When activated, the client application displays the nurse station census and pending notifications can be reviewed and acknowledged by nurses and medical receptionists.

Implementation in the surgical ICU

The system has been piloted in the surgical ICU, a 18-bed unit for trauma and post-operative patients. SICU is the busiest unit in the hospital both in terms of acuity of patients and rate of transfer to and from other locations (ER, operative rooms, radiology). As orders for SICU patients can be written from various locations throughout the hospital, and sometimes by different teams for the same patient, the reliability of the electronic notification is crucial. A missed or delayed order could have serious consequences.

Three workstations at the central medical receptionist's desk were configured to monitor the whole unit. Twelve other workstations, located in the patients rooms, monitored the beds in the same room. Nurses were asked to acknowledge orders on-line. Medical receptionists checked that STAT orders had been seen by the appropriate nurse. The paper notification system was left in place for redundancy during initial system implementation..

RESULTS

The current notification system runs on three low-end Pentium®-based servers and handles the 10,000 notifications generated each day by new or modified orders. It takes 1 to 2 seconds for a notification reaching the GIE to be updated in all subscribing clients. There was no measurable increase in network traffic. Notifications were typically acknowledged by nurses or medical receptionists within 2-10 minutes of being issued (Figure 2). Since the system has been in place, there have been no reports of "missed" or delayed orders, which is an improvement over the previous baseline of 1-2 weekly reported incidents. We have not yet performed a formal comparison with the paper-based notification system.

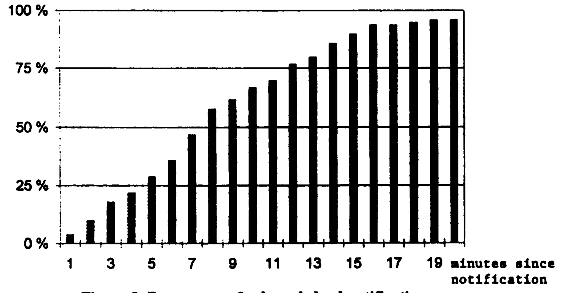

Figure 2. Percentage of acknowledged notifications

Cosmetic changes were requested by the users to adjust the intrusiveness of the visual notification by the client application. Overall, the system was well accepted, and the unit staff felt comfortable that the paper-based notification of new orders could be discontinued.

FUTURE DEVELOPMENTS

As often in such implementation, the system, once in place, generated enthusiastic suggestions in how it could be extended to provide other types of notification, such as new lab results, the advance warning for the arrival of a severe trauma patient, or the fact that a specific ICU is full and in diversion mode. These suggestions will soon be implemented.

One of the advantages of the publish-subscribe model is that, by decoupling the providers and users of information, it enables the independent development of publishing applications, subscribing applications and notification algorithms.

With the basic notification system implemented as a component of the information management architecture, it will be relatively straightforward to add new publishers such as the laboratory results (about 10,000 notifications per day), the ADT transactions (about 3,000 notifications per day), and various alarm sources. Role-specific clients will be developed or embedded in other applications, such as the order entry interface. When they become available, new methods of notification such as e-mail and fax will be added to the palette of resources of the notification algorithms.

Numerous non-clinical applications can make use of this system, including the notification of network technicians of a router failure detected by a monitor, and the ability for developers to subscribe to a list of published resources such as upcoming upgrades or changes in other applications. At a lower level, applications currently relying on the request-reply mechanism to look for updates can be "event-enabled". As it handles data-driven, user-driven and time-driven events and can generate new transactions through the GIE, the notification system can convert complex, linear, data-processing schemes into simpler, iterative algorithms.

As the information technology industry is moving towards distributed event-driven computing, and already provides working systems that use the publish-subscribe paradigm, the future role of this project will not be to duplicate industrial efforts, but to help understand how this technology can be used for improving clinical information systems and enable innovative applications. In essence, the modular approach of our architecture should permit the reuse of publishers, subscribers and notification algorithms, regardless of the transport mechanism, as long as it supports the basic communications model.

The two key areas of further investigation and development are:

- the construction of a uniform hierarchized nomenclature of messages subjects. This will become essential as the scope of the system widens, as the name space of subjects must be shared by all components of the system as well as the applications using it;

- the definition of a common syntax for the messages contents. The use of object technology is a logical way to approach this problem.

CONCLUSION

The publish-subscribe paradigm offers a powerful mechanism for the implementation of an efficient, general clinical notification system. In our design, a notification is conceptualized both as a message to be delivered and an algorithm to deliver it. The fact that the algorithm is controlled by a third party, within the notification system, maintains the decoupling between the providers and consumers of information, while providing necessary capabilities such as acknowledgment and guaranteed delivery.

The role of the notification system goes beyond the distribution of alerts, and can be used as an efficient mechanism of asynchronous communication in a distributed computing environment, and therefore support the wide spectrum of event-driven applications of a clinical information system.

References

1. Pryor RA, Gardner RM, Clayton PD, Warner HR. The HELP system. In: Blum BI ed. Information Systems for Patient Care. New York: Springer Verlag, 1984;109-128.
2. Nguyen LT, Margulies DM. The design of a rule-based clinical event monitor in a multi-vendor hospital computing environment. Symp Comp App in Medical Care. 1992:432-6
3. Hripcsak G, Clayton PD. User comments on a clinical event monitor. Symp Comp App in Medical Care. 1994; 18:636-640
4. Tate KE, Gardner RM, Scherting K. Nurses, pagers, and patient specific criteria: three keys to improved critical value reporting. Symp Comp App in Medical Care 1995; 19:164-168
5. Kuperman GJ, Teich JM, Bates DW, et al. Detecting alerts, notifying the physician and offering action items: a comprehensive alerting system. Proc AMIA Annual Fall Symp 1996; 20: 704-708
6. McDonald CJ, Tierney WM, Overhage JM, et al. The Regenstrief medical record system: 20 years of experience in hospitals, clinics and neighborhood health centers. MD Computing 1992; 4:206-217
7. http://www.omg.org/
8. Stead WW, Borden RB, Boyarsky MW, et al. A system's architecture which dissociates management of shared data and end-user function. Symp Comp App in Medical Care 1991; 475-480
9. Giuse DA, Mickish A. Increasing the availability of the computerized patient record. Proc AMIA Annucal Fall Symp. 1996; 20:633-637
10. Geissbuhler A, Miller RA. A new approach to the implementation of direct care-provider order entry. Proc AMIA Annual Fall Symp. 1996;20:689-693.